

FIG. 1

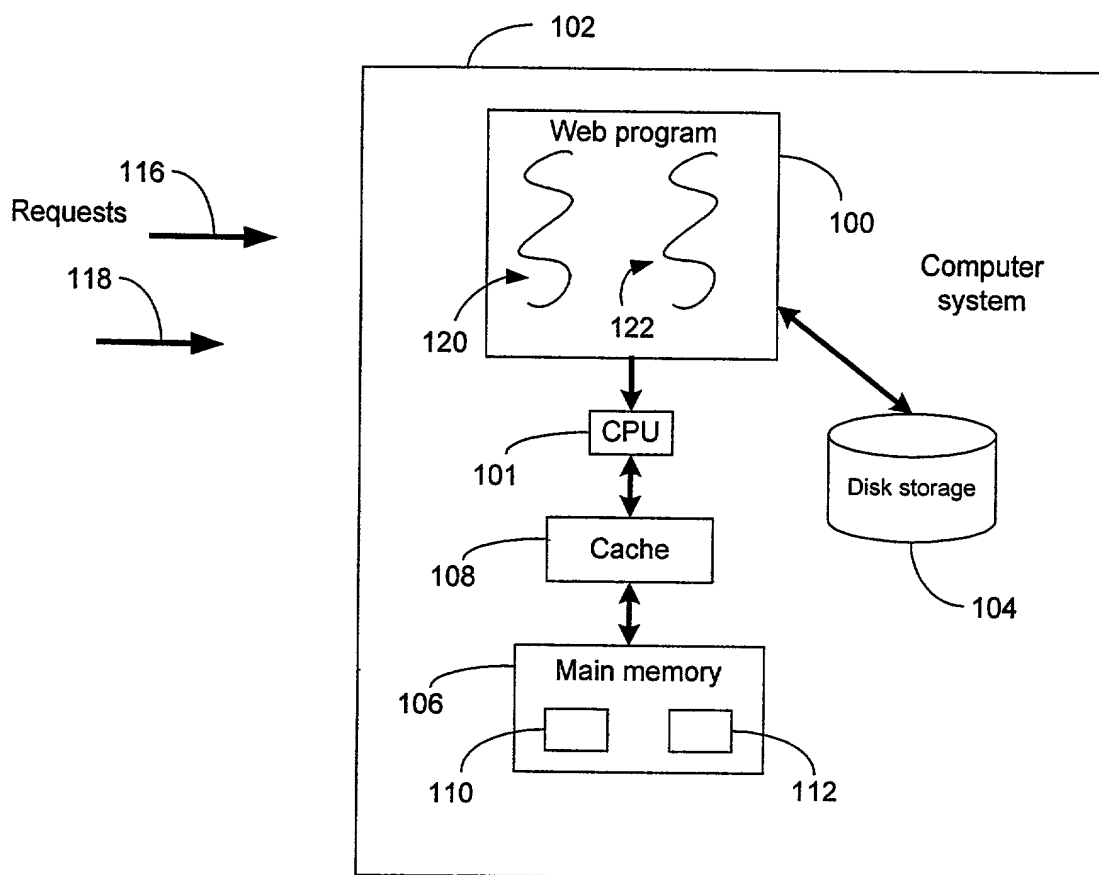


FIG. 2

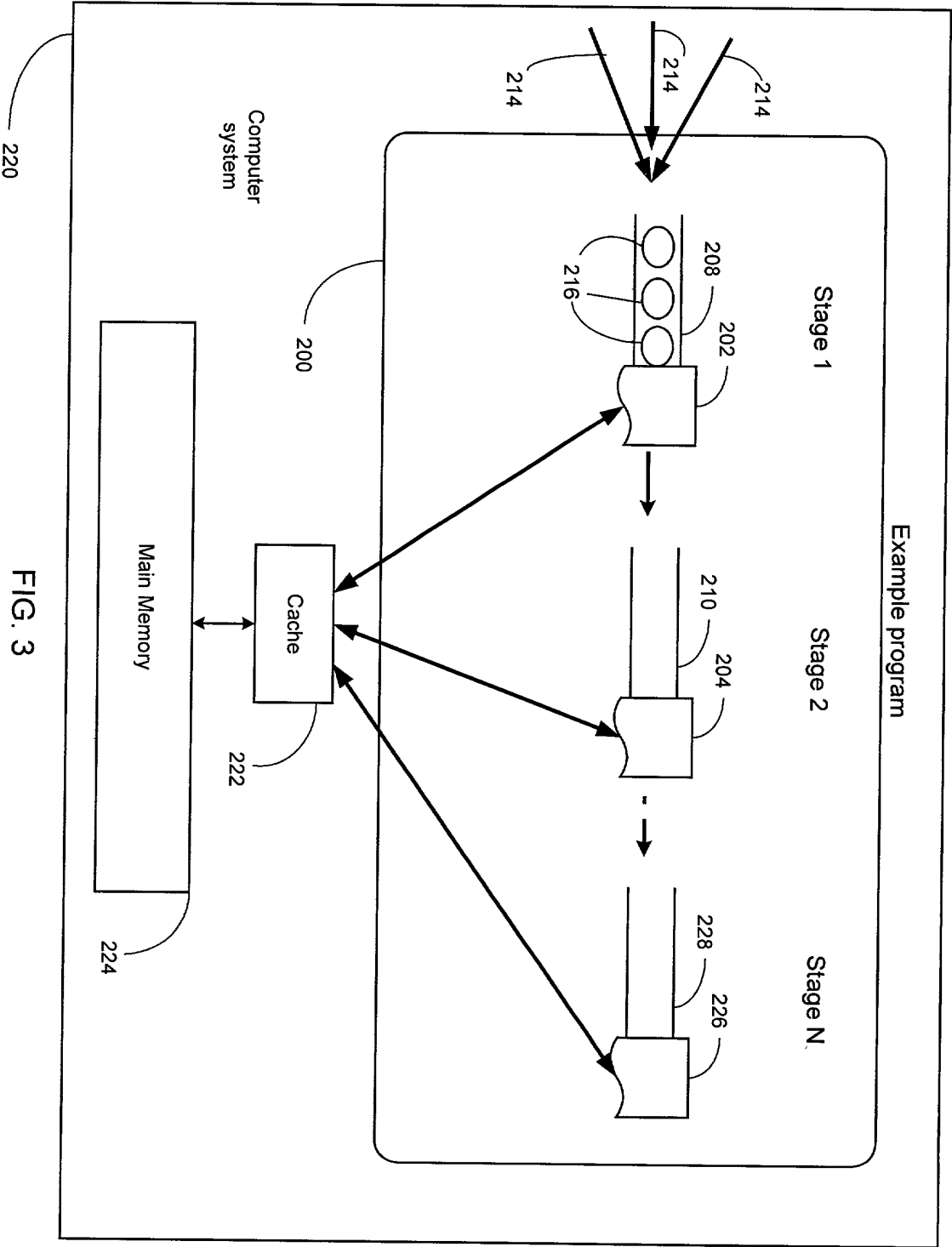


FIG. 3

097963-0000004

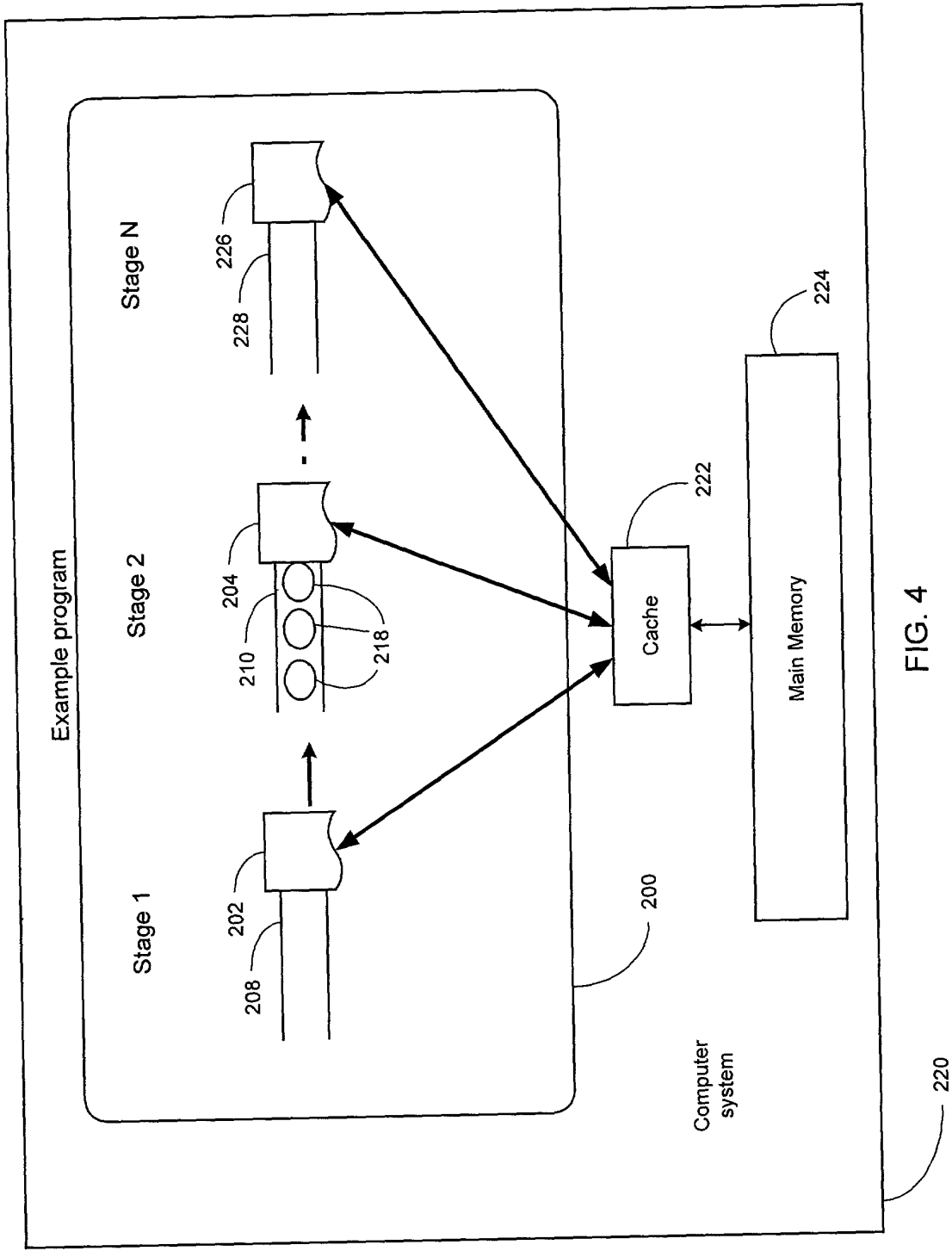


FIG. 4

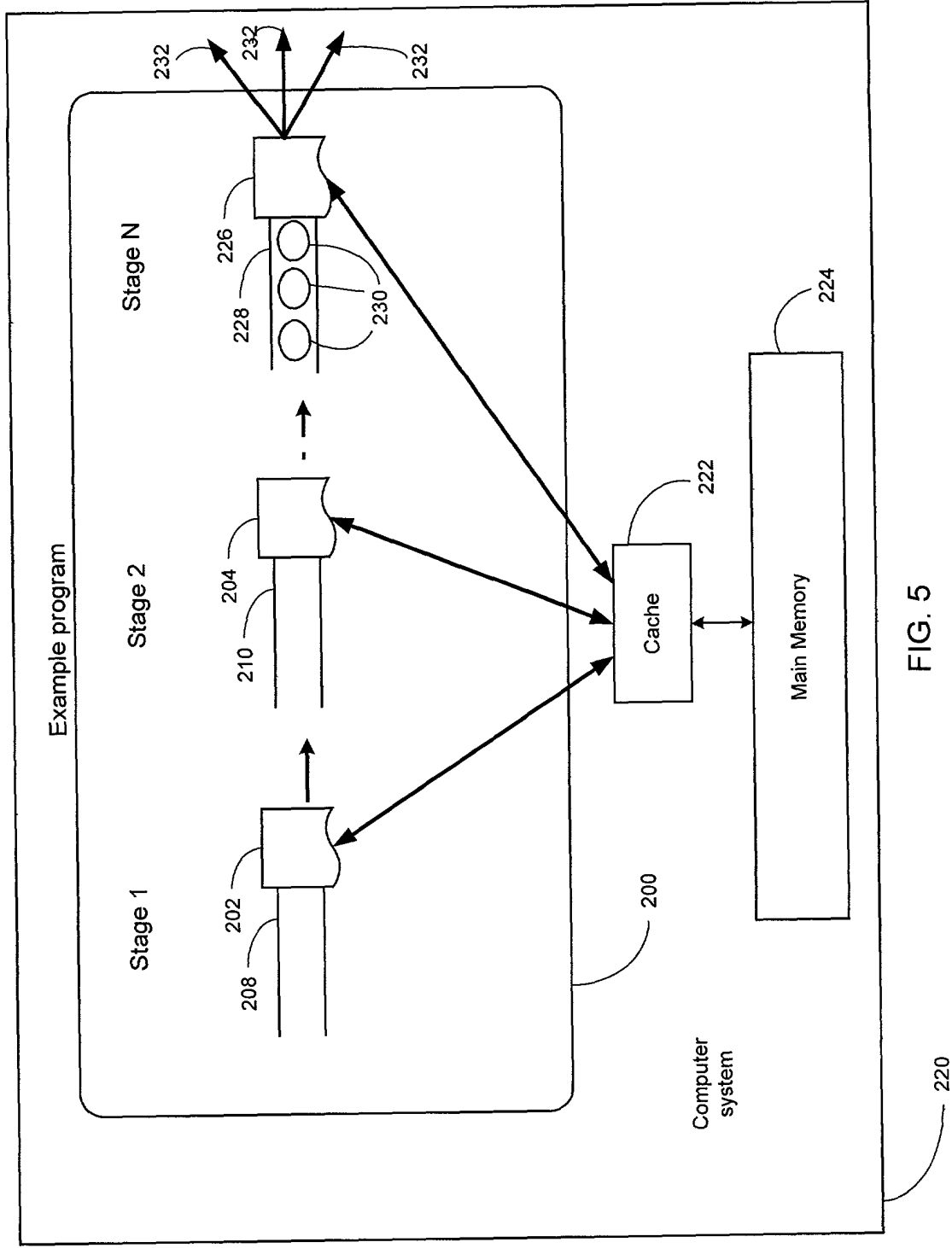


FIG. 5

Stage associated
with subtask 300a

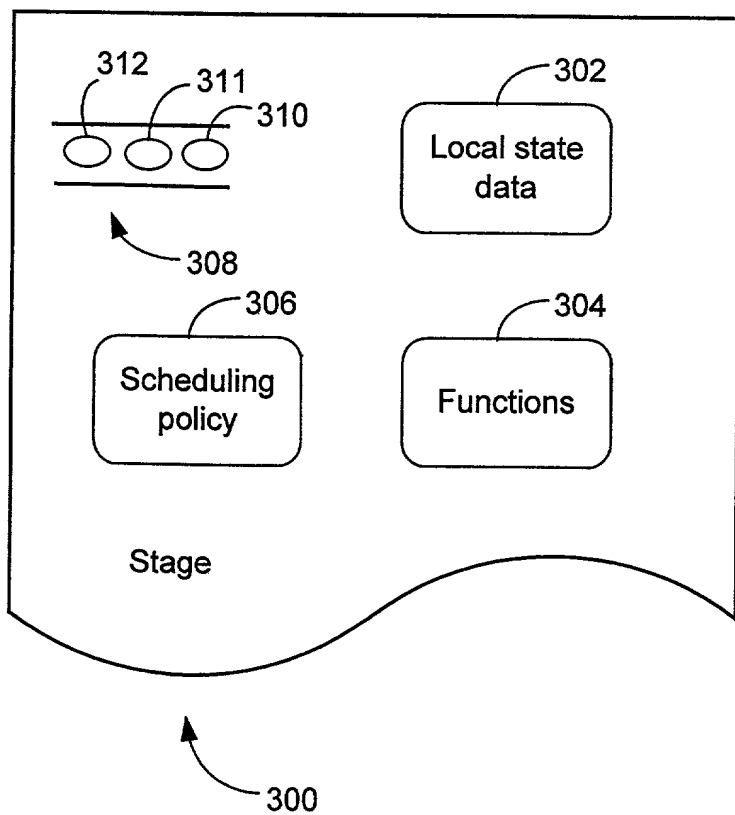


FIG. 6

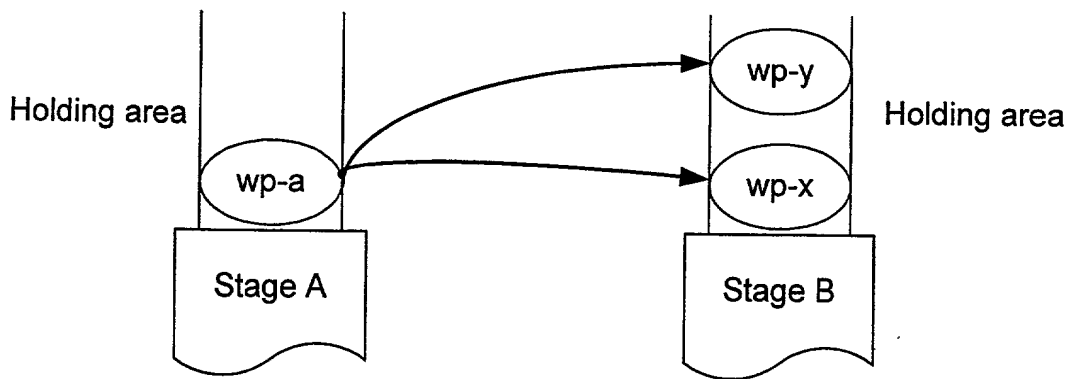


FIG. 7a

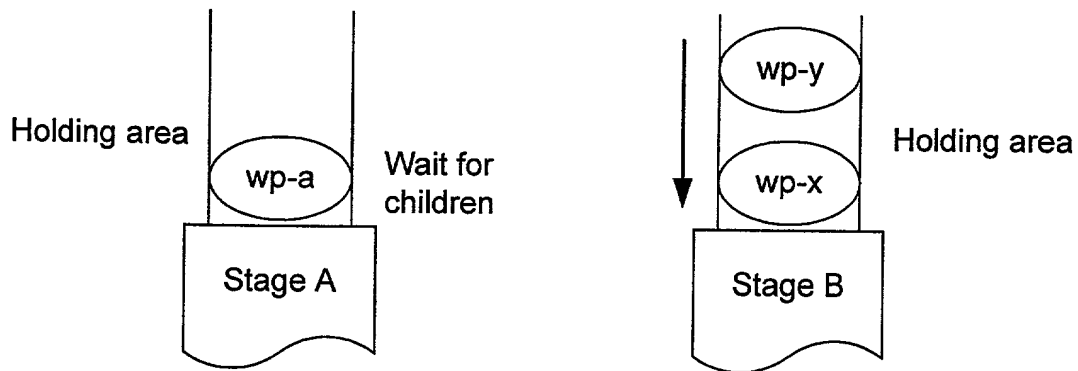


FIG. 7b

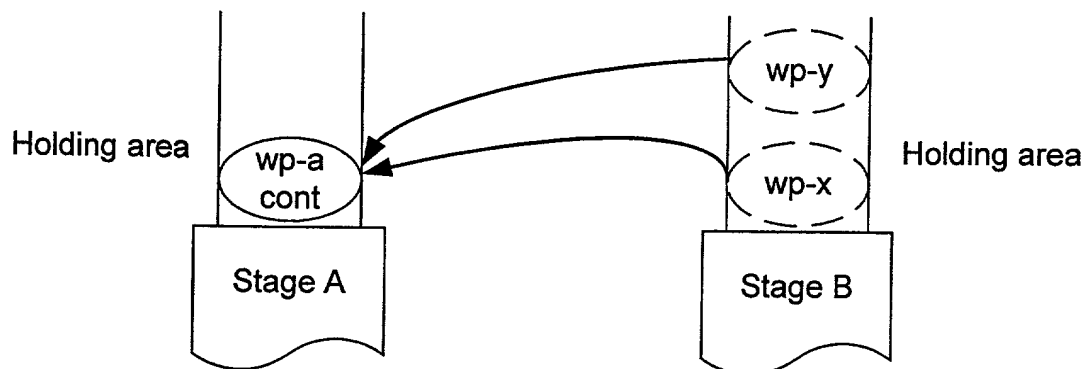
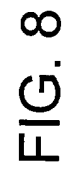


FIG. 7c



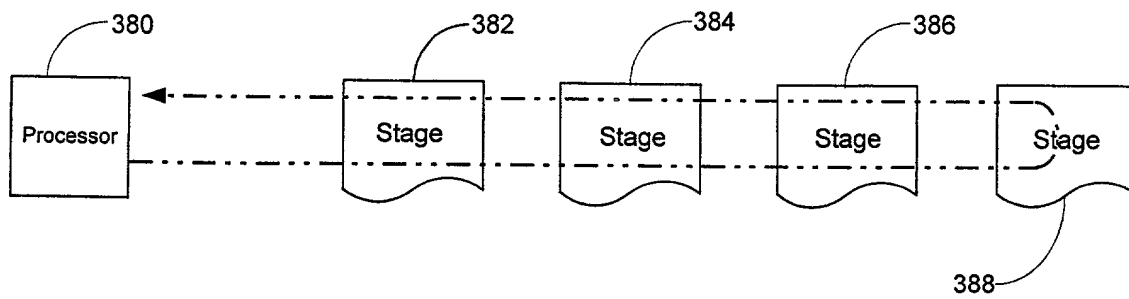


FIG. 9

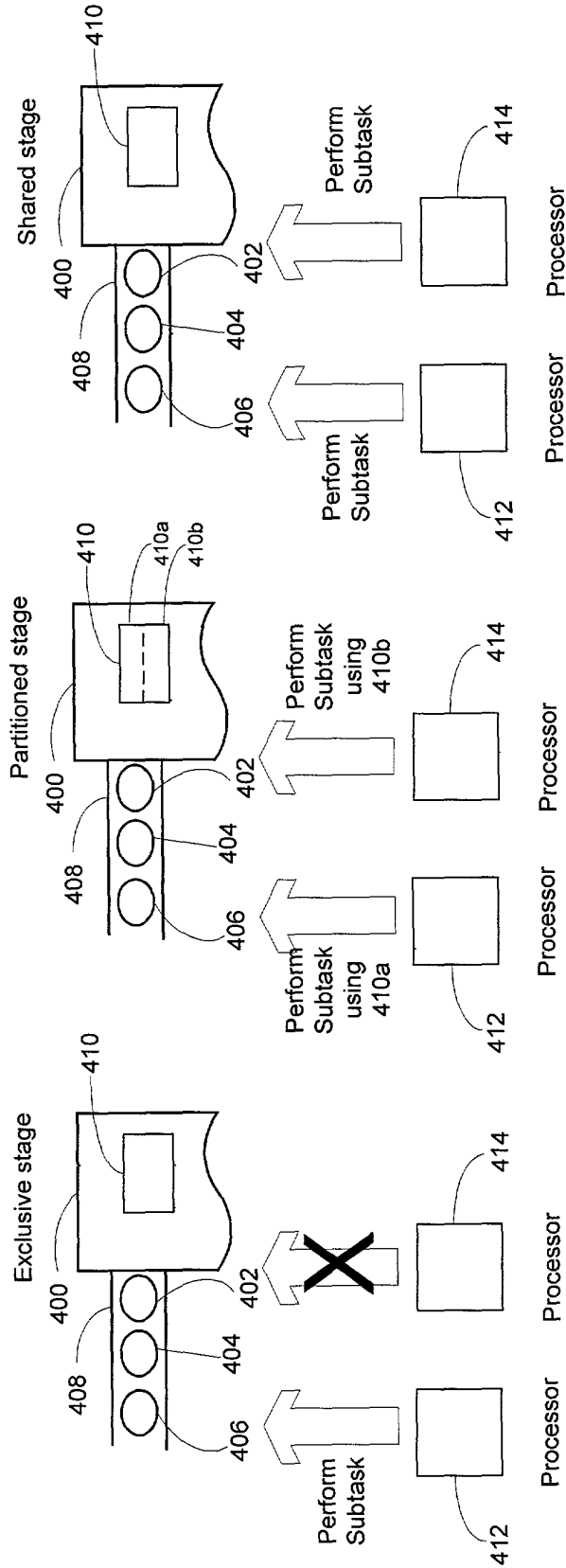


FIG. 10

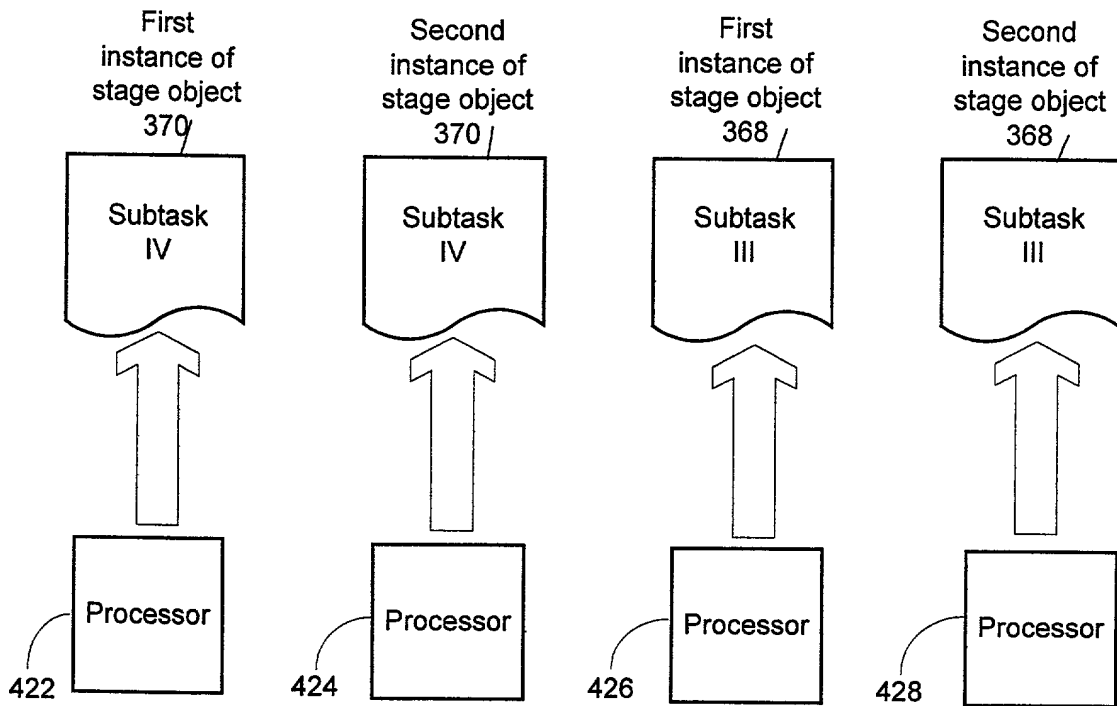


FIG. 11

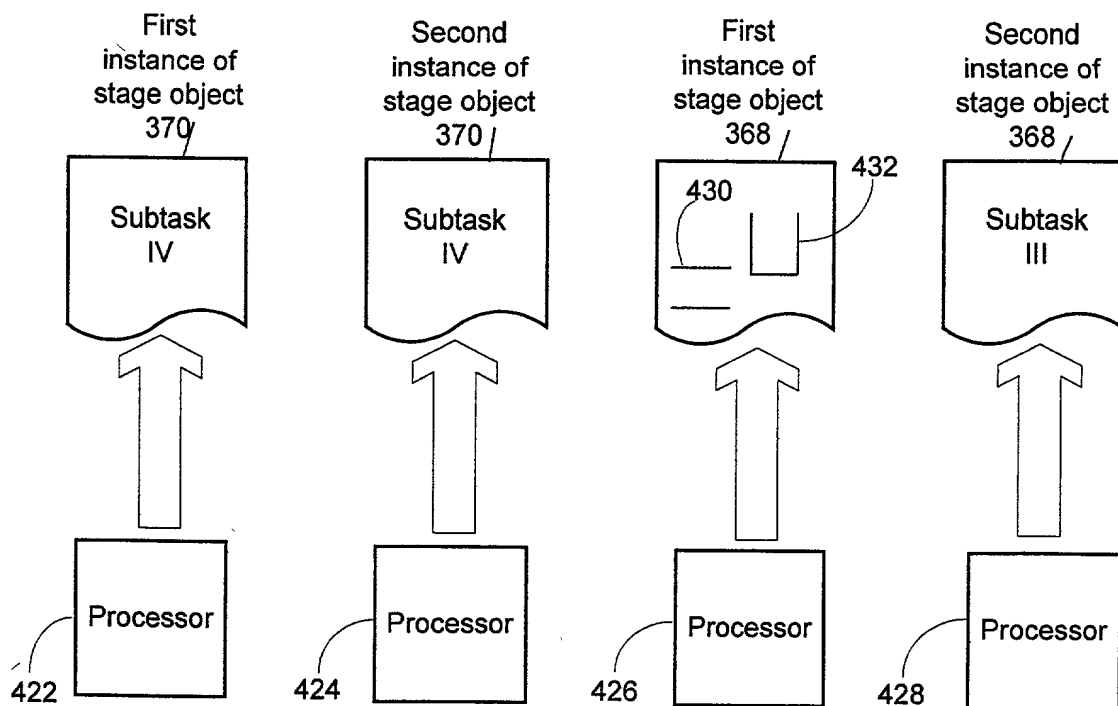


FIG. 12

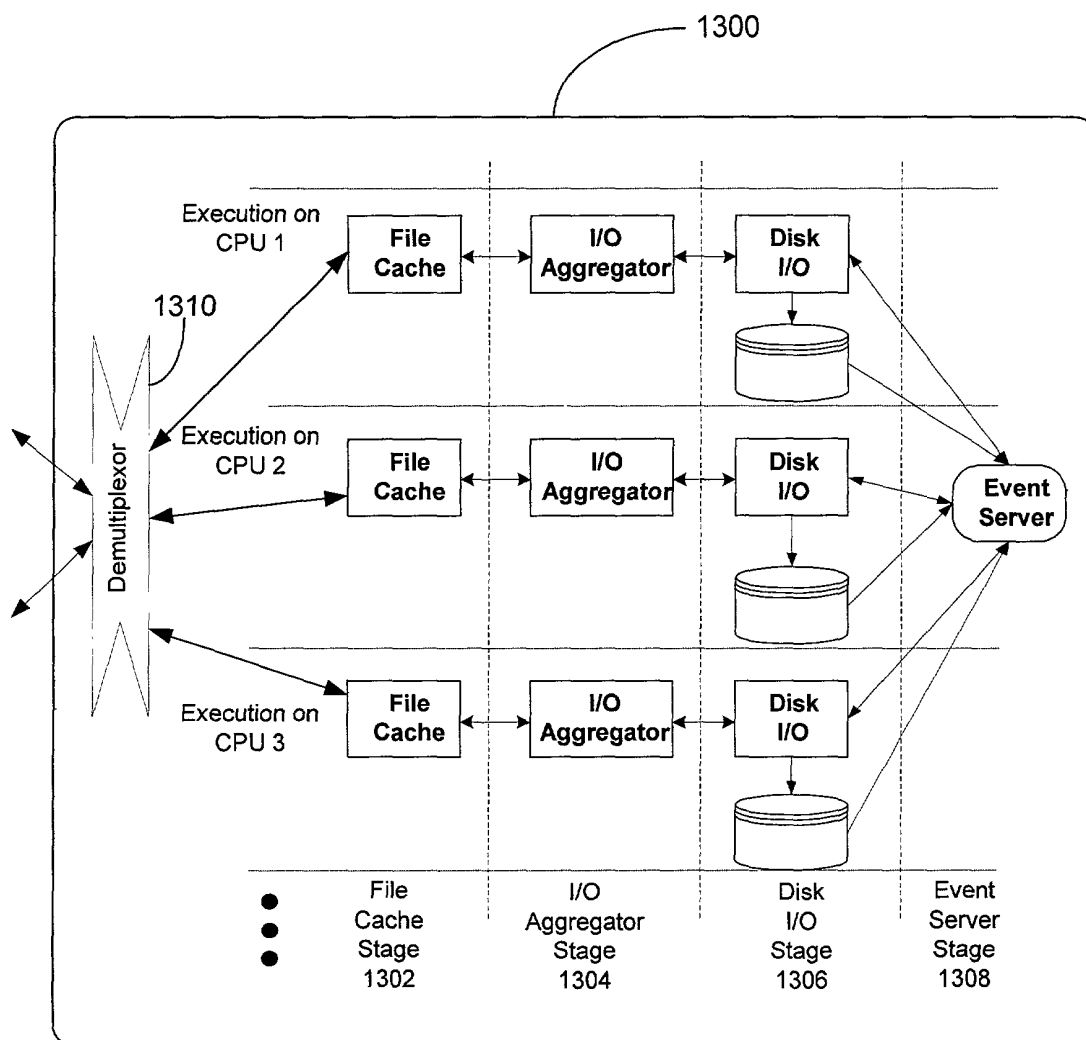


FIG. 13

```
const int MaxChildren= 4;

class MY_STAGE : public STAGE
{
public:
    MY_STAGE() :
        STAGE ("HelloWorldStage", ExclusiveStage, False, 0, 0, DefaultTimer, True, StageBatchSize)
    { }
};

MY_STAGE MyStage;          // Instance of stage used to execute operations

class MY_PACKET : public CLOSURE<MY_PACKET>
{
public:
    RESULT<INT> Children[MaxChildren];
    int Number;

    MY_PACKET() : { }

    MY_PACKET(int NewNumber) : Number(NewNumber) {printf("Creating new child %d\n",
Number);}

    // Parent Work Packet:
    ACTIONS StartChildren() {
        // Create the children and waiting for them to complete
        for (int Count=0; Count < MaxChildren; Count++) {
            new(NewChild, &MyStage, NoPartitionKey, NoSessionKey, &Children[Count]) MY_PACKET(Count);
        }
        return WaitForChildren(WakeAfterSleep);
    }

    // Parent Continuation:
    ACTIONS WakeAfterSleep() {
        // Make sure all children are done
        int Total = 0;
        for (int Count=0; Count < MaxChildren; Count++) { Total += Children[Count]; }

        printf("\nAll children have finished.\n\n");

        printf("The total of 0 + 1 + 2 + 3 = %d\n\n", Total);

        // Quit
        return Complete();
    }

    // Child Work Packet:
    ACTIONS NewChild() {
        // Code for a child, which just returns its index
        RESULT<int> Result = Number;
        printf("New child %d is now running\n", Number);
        return Complete(&Result);
    }
};
```

FIG. 14